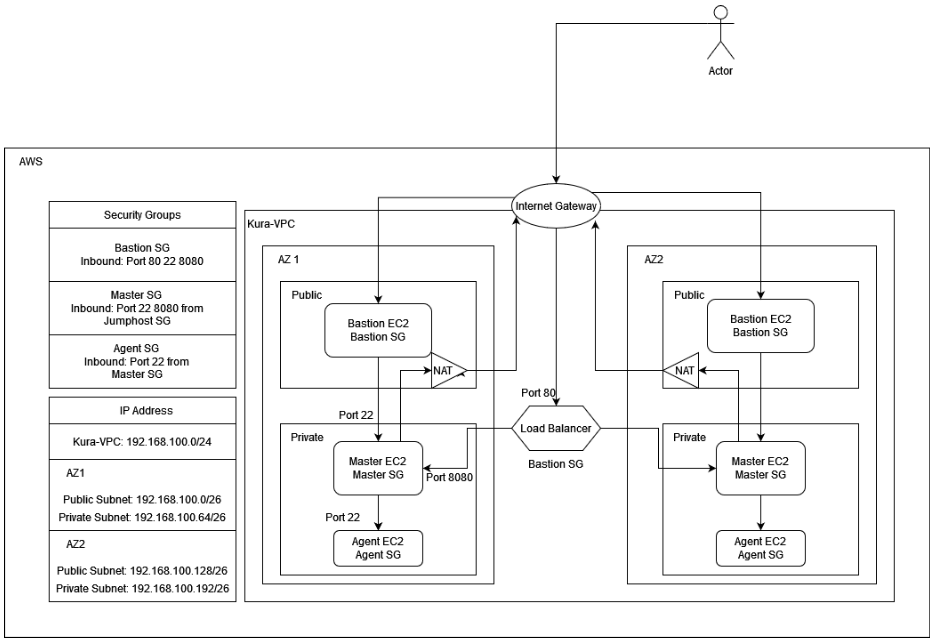
Deployment 5 - Resiliency with AWS

Kenneth Tan

Goal: Architect resilient infrastructure for Jenkins with AWS



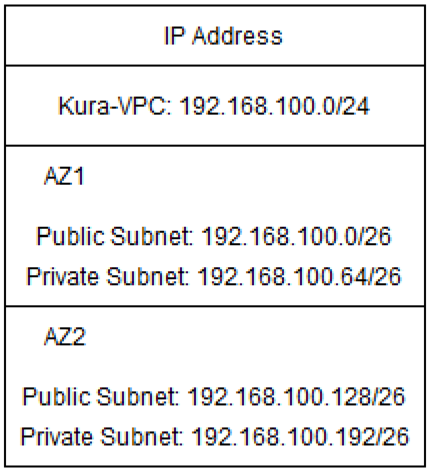
VPC

In AWS there is a service called VIrtual Private Cloud or VPC that allows for you to create your own network. It provisions resources from a datacenter server for whatever compute instances you want to create within your personal network.

Subnetting

Subnetting is splitting a larger network into smaller ones or subnets or subnetworks.
Subnet ip ranges used for projects. We do this using CIDR blocks or ranges of IP addresses for each subnetwork so each compute instance included in the subnet has a set of IP addresses reserved for them.

Below are the IP addresses used for the VPC I created:



IP Address
Kura-VPC: 192.168.100.0/24
AZ1 Public Subnet: 192.168.100.0/26 Private Subnet: 192.168.100.64/26
AZ2 Public Subnet: 192.168.100.128/26 Private Subnet: 192.168.100.192/26

Here I created 2 pairs of public and private subnets. Public subnets are generally interfacing with internet traffic and can be accessed via the internet. Private subnets,however, are used to host applications and to keep your source code safe from end users because they are not internet facing or made public.

Availability Zones

Within AWS when you create a VPC, you are able to select Availability Zones or geological locations for where your network will live. The purpose of this is to add in a layer of fault tolerance or resiliency to for the application you want to serve. In the event that there was something wrong with the server rack where our compute instance was provisioned or if the entire datacenter goes down, we can be assured that our application can still run because there is a copy of it in another location that is far from the original location.

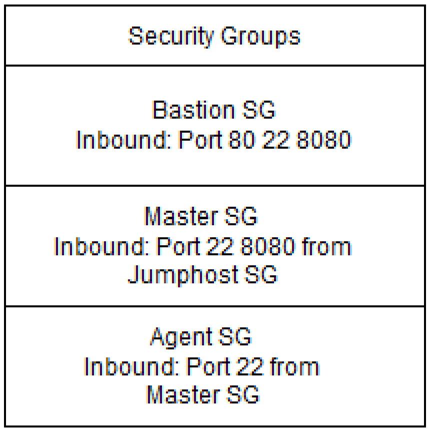
This also provides additional availability. If we have a high amount of traffic coming into our application, instead of overloading one compute instance with all of the traffic, we can split it up between our availability zones.

Load Balancer

You can create an application load balancer to achieve this. The load balancer will take in the internet traffic and redirect it to the appropriate instance. A target group is a feature in AWS Application Load Balancers that will let the load balancer know to which instances to forward the traffic to and to which port.

Security Groups

In order to secure our network and to make sure that the traffic, we can utilize security groups. Security groups act as a type of firewall, only allowing traffic from certain sources. This prevents the internals of our application and network from being too open and secures that the traffic comes from where we expect it.



Security Groups
Bastion SG Inbound: Port 80 22 8080
Master SG Inbound: Port 22 8080 from Jumphost SG
Agent SG Inbound: Port 22 from Master SG

Jenkins

For this particular project, I used this architecture to host Jenkins in a private EC2 instance. The load balancer will take http traffic from port 80 and redirect it to port 8080 on the EC2 instances with jenkins.

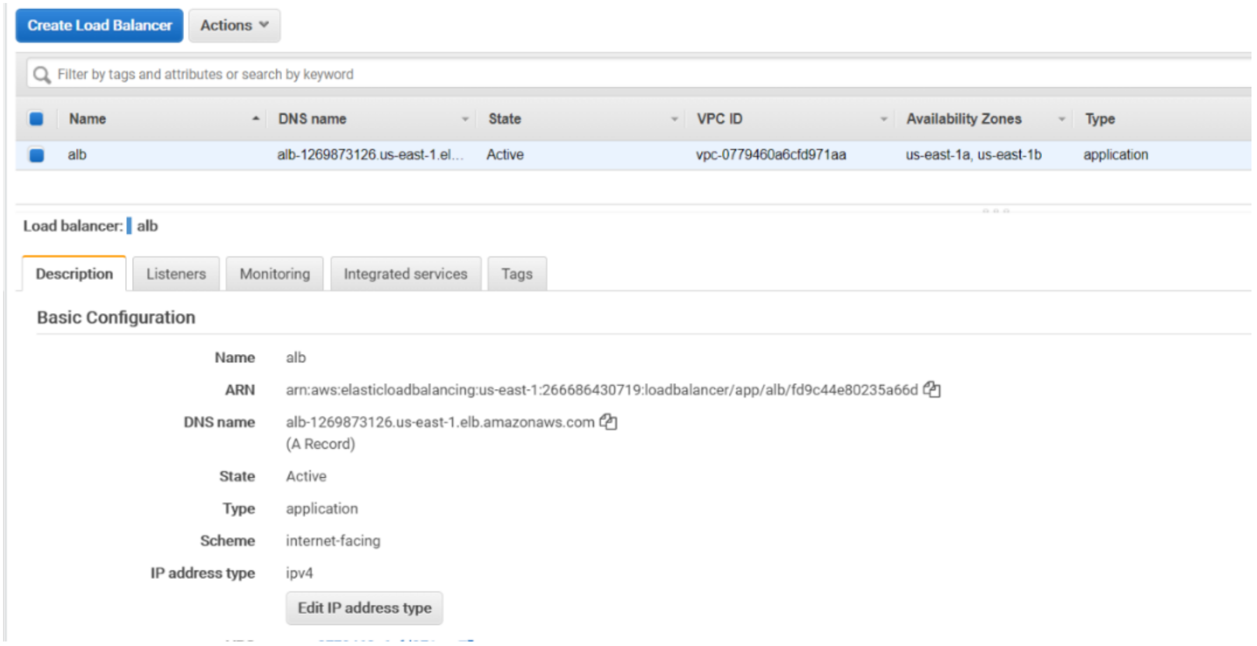
Jenkins was configured to build on a separate instance called an agent. The EC2 with Jenknins running is referred to as the Controller or Master. This is because the purpose of Jenkins on this EC2 is to delegate building and other tasks to the agent. This agent does not have Jenkins installed but will have its computing resources used for whatever processes that jenkins will delegate to it.

To install Jenkins on the Master EC2, you can run this script to do it for you when you first SSH into the EC2 from the Bastion.

```
#!bin/bash

Sudo apt update
Sudo apt upgrade -y
sudo amazon-linux-extras install java-openjdk11 -y
sudo amazon-linux-extras install epel -y
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
sudo yum upgrade -y
sudo yum install epel-release java-11-openjdk-devel -y
sudo yum install jenkins -y
sudo systemctl start jenkins
```

If the architecture is set up properly then you can access the Jenkins web UI with your load balancer's DNS name



Name	DNS name	State	VPC ID	Availability Zones	Type
alb	alb-1269873126.us-east-1.el...	Active	vpc-0779460a6cfd971aa	us-east-1a, us-east-1b	application

Load balancer: alb

Basic Configuration

Name	alb
ARN	arn:aws:elasticloadbalancing:us-east-1:266686430719:loadbalancer/app/alb/fd9c44e80235a66d
DNS name	alb-1269873126.us-east-1.elb.amazonaws.com (A Record)
State	Active
Type	application
Scheme	internet-facing
IP address type	ipv4

When you do, download the recommended plugins and create a username and password.

When you see the homepage, you can create your agent by clicking on the shield in the top right.

Select a new node then create a name for the node and select permanent agent. Create a name and description.

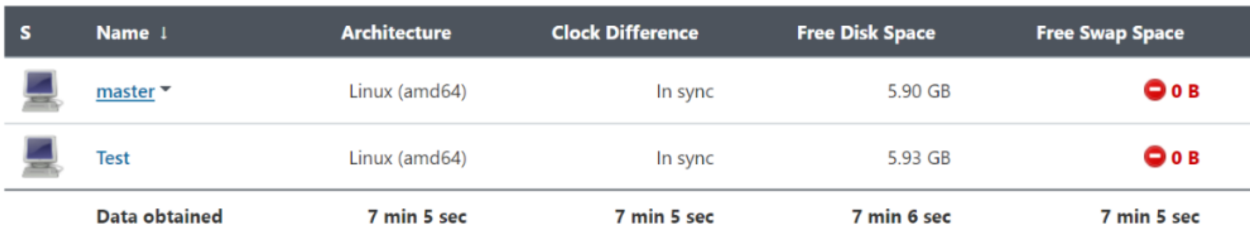
Enter 2 for executor.This enables up to 2 processes to run on the node at once. Enter /home/ec2-user/jenkins for remote root directory or /home/ubuntu/ if you created the EC2 with Ubuntu.

Create a label. This is how the pipeline will call this agent node. Select use this node as much as possible because this is the only agent in this case.

Select launch agent via ssh. Enter the private IP address of the Agent. Add SSH credentials
username: ec2-user | key: the private key you used to ssh into the agent
Username: ubuntu if you used ubuntu

Press save and then look at the logs to see if your setup was successful. It will take a minute or so for the verification and creation to complete

Returning to the Nodes page should show this:



S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space
	master	Linux (amd64)	In sync	5.90 GB	⛔ 0 B
	Test	Linux (amd64)	In sync	5.93 GB	⛔ 0 B
	Data obtained	7 min 5 sec	7 min 5 sec	7 min 6 sec	7 min 5 sec